Method and System for Generating Recommendations

FIELD OF THE INVENTION

[0001] The present invention relates to the field of Internet applications. In particular, the present invention relates to a method and system for generating product or service recommendations using user input data from multiple domains.

BACKGROUND OF THE INVENTION

[0002] Recommendations are one of the strongest ways that people search for and find what they want. Whether shopping for a camera or trying to find a restaurant, people rely on getting recommendations from a variety of sources such as friends, experts, and publications.

[0003] Research in computer-generated recommendations has been active for decades and has resulted in a large amount of publications and various approaches, as well as many working systems with a range of successes. Computer-generated recommendations can be based on the codification of expertise from various sources. This was evidenced in the 1970s and 80s through the glut of expert system-based applications. The problem with this approach is the effort and difficulty in capturing and maintaining expertise in a computer application.

[0004] An alternative approach that addresses some of the drawbacks of expert systems are machine learning systems where the computer application can improve its own performance by adapting itself based on past performance. With regards to recommendations, this approach is manifested in the form of collaborative filtering algorithms.

[0005] Collaborative filtering bases recommendations on the experiences of other users. When faced with a decision in a particular domain people often consult with friends who have experiences in that domain. Collaborative filtering is a way to build those recommendations across a larger group of users, beyond just friends and acquaintances. The approach takes as input the feedback of these users in the form of reviews, scores, ranks,

votes, etc. This input data is analyzed to find patterns and correlations between users and items that translate into probabilities of success of the recommendations.

[0006] A well known personalized movie recommendation website that uses collaborative filtering is MovieLens (http://www.movielens.com). In this personalized movie recommendation service, thousands of users submit scores for movies based on how well they liked or disliked the movies. Recommendations for a specific user can be made by looking at movies that are similar to movies liked by the user. This provides the recommendations of the type, "people who like this movie also like the following movies." For example, a user who is looking at the movie Jerry Maguire may get a movie recommendation for A Few Good Men. In this case "similar" is based on the analysis of scores across the entire user population who has scored movies that intersect with the movies scored by the specific user requesting recommendations.

[0007] As illustrated in the above example, existing systems in the area of collaborative filtering have focused on making recommendations based on users' input data within a single domain. Since the input data is gathered only from a single domain, for example movies, the recommendations can be generated for items within this domain. Furthermore, the quality of collaborative filtering recommendations is dependent on the amount of user input data that is available. But if there is little data available in one domain, the recommendations generated are less relevant and less confident.

[0008] Therefore, there is a need for a system and method that can address the problems of making recommendations based on data collected from a single domain. In particular, there is a need for a system and method for using user input data from multiple domains in making product or service recommendations.

SUMMARY

[0009] The invention discloses a system and method that uses user input data across multiple domains for generating product or service recommendations. This invention provides multiple improvements over existing systems.

[0010] First, this invention is advantageous for organizations whose business interests cross domains. The recommendations are not limited to a single domain. Instead, user events from different domains can be leveraged for producing recommendations in any

of those domains. For example, by analyzing a user's feedback across shopping, news and movies, the system can recommend skateboarding movies to the user who had shopped for skateboards or read news articles about skateboarding.

[0011] In addition, this invention allows for access to a greater amount of user input data which in turn improves the quality of recommendations. In the skateboarding example above, the amount of data solely within the shopping domain with regard to skateboarding might be too minimal to allow for quality recommendations. This might also be true within the news and movies domains, when looked at individually. But by allowing for data to be combined across these domains, this pattern of skateboarding correlation can be found and used in recommendations that otherwise would not be possible.

[0012] A method for generating recommendations across multiple product or service domains includes collecting user events across a plurality of product or service domains in a database, receiving a triggering event for recommendations, analyzing the user events to formulate correlations between the user events in the database, and generating recommendations in response to the triggering event in accordance with the correlations between the user events in the database.

[0013] A system for generating recommendations across multiple product or service domains includes a plurality of domain servers for handling user events and for interfacing with users via the Internet, a database for storing the user events, and a recommendation engine. The recommendation engine further includes one or more computer programs containing instructions for collecting the user events across a plurality of product or service domains in the database, receiving a triggering event for recommendations, analyzing the user events to formulate correlations between the user events in the database, and generating recommendations in response to the triggering event in accordance with the correlations between the user events in the database.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The aforementioned features and advantages of the invention as well as additional features and advantages thereof will be more clearly understood hereinafter as a result of a detailed description of embodiments of the invention when taken in conjunction with the following drawings.

[0015] Figure 1 illustrates a cross-domain recommendation system according to an embodiment of the present invention.

[0016] Figure 2A illustrates a method for collecting user events in the user database 114 of Figure 1 according to an embodiment of the present invention.

[0017] Figure 2B illustrates a method for analyzing user events in the user database 114 of Figure 1 according to an embodiment of the present invention.

[0018] Figure 3 illustrates a method for generating recommendations of similar items according to an embodiment of the present invention.

[0019] Figure 4 illustrates a method for generating personalized recommendations according to an embodiment of the present invention.

[0020] Like numbers are used throughout the figures.

DESCRIPTION OF EMBODIMENTS

[0021] Figure 1 illustrates a cross-domain recommendation system according to an embodiment of the present invention. The recommendation system includes one or more recommendation servers 102, and one or more clients 104. The recommendation servers 102 interface with the clients 104 via the Internet 103. The recommendation servers further include a plurality of individual domains, for example, shopping domain 106, news domain 108, movies domain 110 and other domains 112.

[0022] A domain is a computer system implemented with different hardware and software for a specific application, such as the shopping application 106, news application 108, and movie application 110. Users' interactions with each particular domain, also referred to as user events, are saved and updated in the user database 114. The user database 114 comprises storage for data collected from each individual domain, for example movies data 116, news data 118, shopping data 120 and other domain data 122. A user 104 may have multiple user interactions with all of the domains. In addition, a domain can be more generally defined as a category of information. For example, different domains can include demographic information (age, race, job, education level, etc.), behavioral information (hours of internet use, variety of websites used, number of computers used to access the

internet, etc.), and computer system information (internet connection capacity, multimedia system capabilities, browser version, etc.).

[0023] The recommendation servers 102 further includes a recommendation engine 126. As will be described in the following sections, the recommendation engine 126 analyzes portions of the user database 114 across domains in response to a specific user event to find correlations across domains for producing recommendations. The recommendation engine 126 processes a data slice 124 from the various domains, for example movie data 116, news data 118, shopping data 120 and other domain data 122. Note that the user events stored in different domains can have different type, value, and structure. The recommendation engine 126 then finds correlations between the user events, and formulates its recommendations based on the correlations between the user events.

Collecting User Events

[0024] Recommendations from collaborative filtering are derived from analysis of user behavior. This user behavior is represented in the user database 114 as user events. Storing these user events from multiple domains is more complex than from just a single domain because the data format from each domain may be different. The user events that are stored in the user database 114 are diverse given the diversity of the domains from which they are collected. Types of events from the shopping domain 106, for example, may include browsing products, purchases, or returns. In news domain 108 event types may include viewing titles, viewing summaries, or viewing entire news articles. In movie domain 110 event types would include scoring and reviewing movies, or purchasing online movie tickets.

[0025] In addition to the variety of types of events, there is a variety of event values for a given event:

- *numerical values* – scores and ranks for movies, songs, and etc.
- *ordinal values* – text values that have an ordered scale, for example, high, medium, low
- *free text* – reviews, opinions
- *nominal values* – everything not included above, usually a set of one or more labels, for example, (on, off), (emailed, messaged, phoned)

[0026] The user database 114 supporting cross-domain recommendations is capable of storing this variety of event values. In addition, the user database 114 for storing user

events from multiple domains is organized across multiple domains as illustrated in Figure 2B below. In many organizations, the domains are managed by different teams of individuals. The aggregation of data from multiple domains by the disclosed recommendation system is coordinated and synchronized across diverse teams of people in different geographical locations. Furthermore, any ongoing changes in the recommendation system are dynamically updated. In other words, changes in all domains are accommodated by this centralized user database 114. The set of domains, products, services, users, and attributes of the user events may be changed constantly. As discussed below, the invention provides support for this dynamic aspect of the user database.

[0027] Figure 2A illustrates a method for collecting user events in the user database 114 of Figure 1 according to an embodiment of the present invention. A user event is described by the following input parameters:

- *userid* – unique identifier for a specific user
- *domain* – business, property or system generating the event, e.g., shopping, news, movies
- *itemid* – unique identifier for a specific item or product within the domain
- *event type* – the type of event being stored, e.g., purchase, score, review
- *event value* – the user's input for the given domain, item, and event type

[0028] The method starts in step 202 and thereafter moves to step 204 where the method receives a user event to be stored and determines if this is a valid user event. If it is not a valid event, then the method determines whether it should be rejected or the database should be extended. If the event is not rejected then the database is updated to reflect the new user event.

[0029] In step 206, an evaluation is made as to whether the event's *domain* is valid. If the *domain* is valid (206_yes), then the method continues in step 212. If not (206_no), in step 208 the recommendation system's configuration is checked to determine if the database can be updated dynamically when new domains are received. If dynamic *domain* updates are not enabled, the method moves to step 238 where the event is rejected. Else if dynamic *domain* updates are enabled, the *domain* is added to the set of valid domains in step 210 and the method continues in step 212.

[0030] In step 212, the method checks if the *event type* is valid. If it is valid (212_yes), then the method continues in step 218. If the *event type* is not valid (212_no), the

method checks if dynamic *event type* update is enabled in step 214. If dynamic *event type* update is not enabled (214_no), the method moves to step 238 and the event is rejected. Else if dynamic *event type* update is enabled (214_yes), the *event type* is added to the set of valid event types in step 216 and the method continues in step 218.

[0031] Next in step 218, the method checks if the *event value* is valid. If it is valid (218_yes), then the method continues in step 224. If the *event value* is not a valid (218_no), the method checks if dynamic *event value* update is enabled in step 220. If dynamic *event value* update is not enabled (220_no), the method moves to step 238 and the event is rejected. Else if dynamic *event value* update is enabled (220_yes), the *event value* is added to the set of valid event values in step 222 and the method continues in step 224. It is advantageous to have the flexibility to accept the variety of event values dynamically. For example, for the domain "Personals" and the *event type* "Method of Contact" the known values might be "(email, voicemail, pager)". As the Personals product changes, a new method of contacting other people may be added, such as "webcam". As users take advantage of this new feature, new user events will be generated with the value "webcam" for "Method of Contact". If dynamic *event value* update is enabled, the system can accept this user event and continue. This method allows the system to automatically grow and change along with the individual domain applications and teams.

[0032] In step 224, the method checks if the *itemid* is valid. If it is valid (224_yes), then the method continues in step 224. If the *itemid* is not a valid (224_no), the method checks if dynamic *itemid* update is enabled in step 226. If dynamic *itemid* update is not enabled (226_no), the method moves to step 238 and the event is rejected. Else if dynamic *itemid* update is enabled (226_yes), the *itemid* is added to the set of valid *itemids* in step 228 and the method continues in step 230. Similar to the flexibility described above for event values, this flexibility for dynamically updating *itemid* allows the system to automatically handle changes in product inventory that are inevitable in most business applications. In some applications, the ability to disable dynamic *itemid* update is essential. For example, if it is desired to only generate recommendations on a subset of items, then only those items can be allowed in the list of valid itemids. Disabling dynamic *itemid* update prevents the undesired items from creeping into the pool of candidate recommendations. In order to maintain the uniqueness of itemids across all domains, itemids are checked for validity only within the *domain* from which it is originated. This allows each *domain* (team, property,

application) to maintain their own processes for item identification without concern of name space conflict with other domains.

[0033] In step 230, the method checks if the *userid* is valid. If it is valid (230_yes), then the method continues in step 236. If the *userid* is not valid (230_no), the method checks if dynamic *userid* update is enabled in step 232. If dynamic *userid* update is not enabled (232_no), the method moves to step 238 and the event is rejected. Else if dynamic *userid* update is enabled (232_yes), the userid is added to the set of valid *userids* in step 234 and the method continues in step 236. Similar to the flexibility described above for *itemids*, the ability to automatically expand the database to store events for new users is essential in most Internet applications where the user population changes everyday. However, the flexibility to restrict these updates can also be beneficial, for example in the case where users' events should not be stored unless they have agreed to the terms of service.

[0034] In step 236, the user event is stored to the user database 114, and the method ends in step 240.

Analyzing User Events

[0035] Figure 2B illustrates a method for analyzing user events in the user database 114 of Figure 1 according to an embodiment of the present invention. As shown in Figure 2B, the user database 114 is represented as a multidimensional array where each user is represented in one or more rows with each row containing a particular user event.

[0036] Using the skateboarding example above, the first user event collected is that User 1 has viewed an entire news article about skateboarding. The *event value* in this case is a nominal value ("yes"). The second event is User 1 has rated a skateboarding movie with a rating of 80. The *event value* is an ordinal value, in this case an integer between 0 and 100, inclusive. The third event is User 1 has subsequently written a review of this movie. The *event value* in this case is free text. The fourth event is User 1 has purchased a skateboard. In this case the *event value* is an ordinal which reflects the amount paid for the item. And subsequently the fifth event shows a rating for these skateboarding items. The *event value* in this case is an ordinal, "high" on the scale of "low", "medium", and "high".

[0037] There are many collaborative filtering algorithms that can be employed to find correlations between user events. In one embodiment of the present invention, the

Affinity Engine described in U.S. patent application, Serial Number 10/417,709, filed April 16, 2003, entitled Affinity Analysis Method and Article of Manufacture, is used to find correlations between user events. This U.S. patent application, Serial Number 10/417,709, is expressly incorporated herein by this reference. The Affinity Engine processes portions of the user database 114 across multiple domains collected over the entire user population to determine the correlations between user events, such as between the user events of User 1 and other users in the user database. Correlation values are assigned to indicate the weight of relationship between canonical user events, and the correlation values are stored in a similarity database. These correlation values are then used to generate recommendations to subsequent users who demonstrate similar user behaviors. For example, if User 2 has also viewed the same entire news article about skateboarding, rated and written a review of the same skateboarding movie, then a recommendation may be generated to User 2 for purchasing certain skateboards based on the collective canonical user events exhibited by other users in the user population, such as the canonical user events of User 1. Note that the disclosed invention is independent of the collaborative filtering algorithm used for computing the correlation values between user events. Recognizing that certain types of filtering algorithms may be more suited for filtering certain types of user events, the disclosed invention supports integration of any collaborative filtering algorithm as a plug-in component to the recommendation system.

Recommending Similar Items

[0038] Similarities between items provide possible criteria for making recommendations. Often people find books to read that are considered similar to other books they have enjoyed. This extends to other domains: movies, songs, clothes, games, and other products and services. Items are considered similar if they are alike, or more explicitly if they have similar values across their attributes. For example, in the case of books the attributes may be author, fiction or non-fiction, genre, and character development. Different approaches exist for computing recommendations based on item similarities.

[0039] In one embodiment of the present invention as illustrated in Figure 3, the method provides a way to use these item similarities in the context of cross-domain recommendations. A triggering event, such as a user request, for similar item recommendations comprises the following input parameters:

- *domain* – business, property or system containing the item for which similar item recommendations are being requested, e.g., shopping, news, movies
- *itemid* – unique identifier for a specific item or product within the domain
- *threshold* – the minimum level of similarity a recommended item must have
- *minitem* – the minimum number of recommended items desired
- *maxitem* – the maximum number of recommended items desired
- *src_domains* – the domain(s) from which recommended items should belong

[0040] The method finds similar items across all desired domains and returns a list of recommended items according to a set of predefined constraints.

[0041] Figure 3 illustrates a method for generating recommendations of similar items by the recommendation engine 126 according to an embodiment of the present invention. The method starts in step 302 and thereafter moves to step 304 where the method receives a triggering event, such as a user request, for recommendations of other items similar to a designated item. In step 306, a determination is made as to whether the designated item exists in the user database 114. If the designated item does not exist (306_no), then the method ends in step 334 and no recommendations are generated. Else if the designated item does exist (306_yes), the method continues in step 308. Determination of whether the designated item exists is based on whether there are any items considered similar to this designated item. The lack of similar items has two possible causes: 1) this is a new item that did not previously exist; 2) this designated item has existed prior to the request, but not enough user events are associated with this designated item on which to base recommendations.

[0042] Next in steps 308 through 312, a set of unrestricted similar items is retrieved; this set is also referred to as the first list of recommendations. The set of unrestricted similar items depends on whether the *src_domains* input parameter is specified. If *src_domains* is specified (308_yes), the set of similar items is drawn from only the domains indicated by the *src_domains* in step 310. If *src_domains* is not specified (308_no), the set of similar items is drawn from all domains in step 312.

[0043] Given the first list of recommendations of similar items, a predefined set of constraints, including *domain*, *threshold*, *minitem* and *maxitem*, are applied to improve the cross-domain recommendations. In step 314, a determination is made as to whether the number of items in the first list of recommendations is greater than the predefined *minitem*. If the outcome of the determination is negative (314_no), the first list of recommendations

is returned in step 332. Else if the outcome of the determination is positive (314_yes), the method continues in step 316.

[0044] In step 316, a second list of recommendations is formed from the first list of recommendations. The second list of recommendations is a subset of the first list of recommendations comprising similar items having a correlation value above the predefined *threshold*. Another determination is made as to whether the number of items in the second list of recommendations is greater than the *minitem*. If the number of items in the second list of recommendations is not greater than the *minitem* (316_no), then in step 330, the method selects *minitem* number of recommendations in descending order of correlation value from the first list of recommendations to form a third list of recommendations, and the method returns the third list of recommendations in step 332. In this situation, one or more of the items recommended may have a correlation value below the *threshold*, but they are returned nonetheless to meet the *minitem* constraint. It is advantageous to present to users the number of items requested in many common computer and internet applications. In the alternative, if the number of items in the second list of recommendations is greater than *minitem* (316_yes), then the second list of recommendations is retained in step 318 for further processing in steps 320 to 328.

[0045] In step 320, the *maxitem* constraint is taken into consideration. A determination is made as to whether the number of items in the second list of recommendations is greater than the *maxitem* constraint. If the outcome is no (320_no), then the second list of recommendations is returned in step 332. This path returns a list of recommended similar items that meet or exceed the threshold, and the number of recommended items is greater than the minimum number of items requested but less than or equal to the maximum number of items requested. In the alternative (320_yes), after applying the similarity *threshold* constraint if the number of items in the second list of recommendations is still more than the maximum desired number of items, the method continues in step 322 to improve the second list of recommendations by spreading the recommended items across the set of domains from which they have been drawn.

[0046] In step 322, the candidate items are separated into domain groups in accordance with the predefined *src_domains* parameter. In steps 324 to 328, the method employs a round-robin scheme and traverses each group one at a time (step 324). In step 326, the method selects a recommendation having the highest correlation value to form a

fourth list of recommendations (step 326). One skilled in the art will recognize that other priority schemes may be employed in the selection process. The steps 324 and 326 are repeated until the number of items in the fourth list of recommendations is equal to the predefined *maxitem*. The final outcome is the fourth set of recommendations that are similar to the requested item. This fourth set of recommendations meets the *minitem* and *maxitem* constraints, as well as the similarity *threshold* if enough similar items exist. The set of recommended items also reflect an even selection across the specified domains. In step 332, the fourth list of recommendations is returned. The method ends in step 334.

Recommending Personalized Items

[0047] Recommending similar items can be further improved by taking into consideration the history of a user's personal preferences. Personalized recommendations are made by finding items similar to those items preferred by a given individual in accordance with the history of the user's personal preferences. Figure 4 illustrates a method for generating personalized recommendations according to an embodiment of the present invention. The method finds personalized recommendations across all desired domains and returns a list of personalized recommendations according to a set of predefined parameters. A triggering event, such as a user request, for personalized recommendations comprises the following input parameters:

- *userid* – unique identifier for a specific user
- *threshold* – the minimum level of similarity a recommended item must have
- *minitem* – the minimum number of recommended items desired
- *maxitem* – the maximum number of recommended items desired
- *src_domains* – the domain(s) from which recommended items should belong

[0048] The method starts in step 402 and thereafter moves to step 404 where the method receives a triggering event for personalized recommendations. In step 406, a first determination is made as to whether the user exists. If the user does not exist (406_no), the method ends in step 434 and no personalized recommendations are generated. In the alternative, if the user exists (406_yes), the method continues in step 408. Determination of whether the user exists is based on whether there are any user events in the user database 114. In step 408, a first list of items that the user has shown a proclivity for or has shown preference is retrieved. The first list of items is also at or above the predefined *threshold*.

[0049] These preferred items are used as the basis for finding personalized recommendations. In steps 410 and 412, for each preferred item (step 410), a set of similar items is retrieved (step 412). The method for finding similar items is described above in association with Figure 3. The following input parameters are used in the process for getting similar items: *threshold*, *minitem*, *maxitem*, and *src_domains*. The *itemid* and *domain* input parameters for getting similar items are derived from each item in the preferred item list. In step 412, the union of similar items retrieved for each preferred item is stored in a first list of personalized recommendations for subsequent steps.

[0050] Given the first list of personalized recommendations, a predefined set of constraints, including *domain*, *threshold*, *minitem* and *maxitem*, are applied to improve the cross-domain recommendations. In step 414, a determination is made as to whether the number of items in the first list of recommendations is greater than the predefined *minitem*. If the outcome of the determination is negative (414_no), the first list of recommendations is returned in step 432. Else if the outcome of the determination is positive (414_yes), the method continues in step 416.

[0051] In step 416, a second list of recommendations is formed from the first list of recommendations. The second list of recommendations is a subset of the first list of recommendations comprising items having a correlation value above the predefined *threshold*. Another determination is made as to whether the number of items in second list of recommendations is greater than the *minitem*. If the number of items in the second list of recommendations is not greater than the *minitem* (416_no), then in step 430, the method selects *minitem* number of recommendations in descending order of correlation value from the first list of recommendations to form a third list of recommendations, and the method returns the third list of recommendations in step 432. In this situation, one or more of the items recommended may have a correlation value below the *threshold*, but they are returned nonetheless to meet the *minitem* constraint. It is advantageous to present to users the number of items requested in many common computer and internet applications. In the alternative, if the number of items in the second list of recommendations is greater than *minitem* (416_yes), then the second list of recommendations is retained in step 418 for further processing in steps 420 to 428.

[0052] In step 420, the *maxitem* constraint is taken into consideration. A determination is made as to whether the number of items in the second list of

recommendations is greater than the *maxitem* constraint. If the outcome is no (420_no), then the second list of recommendations is returned in step 432. This path returns a list of recommended items that meet or exceed the threshold, and the number of recommended items is greater than the minimum number of items requested but less than or equal to the maximum number of items requested. In the alternative (420_yes), after applying the similarity *threshold* constraint if the number of items in the second list of recommendations is still more than the maximum desired number of items, the method continues in step 422 to improve the second list of recommendations by spreading the recommended items across the set of domains from which they have been drawn.

[0053] In step 422, the candidate items are separated into domain groups in accordance with the predefined *src_domains* parameter. In steps 424 to 428, the method employs a round-robin scheme and traverses each group one at a time (step 424). In step 426, the method selects a recommendation having the highest correlation value to form a fourth list of recommendations. One skilled in the art will recognize that other priority schemes may be employed in the selection process. The steps 424 and 426 are repeated until the number of items in the fourth list of recommendations is equal to the predefined *maxitem*. The final outcome is the fourth set of recommendations that are personalized to the user's preferred items. This fourth set of recommendations meets the *minitem* and *maxitem* constraints, as well as the *threshold* of correlation value if enough personalized items exist. The set of recommended items also reflect an even selection across the specified domains. In step 432, the fourth list of recommendations is returned. The method ends in step 434.

[0054] The disclosed system for generating recommendations offers a number of improvements over the existing systems. Specifically, the disclosed system generates better and more relevant recommendations by taking into consideration user input data across multiple domains. This capability of generating better recommendations in turn enables better user experience and provides a broader range of products or services to users.

[0055] One skilled in the relevant art will recognize that there are many possible modifications of the disclosed embodiments that could be used, while still employing the same basic underlying mechanisms and methodologies. For example, different sets of constraints, such as demographic information of a user, may be used in improving the recommendations. Different priority schemes may be employed in place of the round-robin

scheme disclosed in one embodiment of the present invention. And different algorithms may be used to compute the correlation values between user input data across multiple domains.

[0056] The foregoing description, for purpose of explanation, has been described with references to specific embodiments. However, the illustrative discussions above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated.